Exploring into the Essence of Choreography*

Qiu Zongyan, Cai Chao, Zhao Xiangpeng, and Yang Hongli
LMAM & Department of Informatics, School of Math., Peking University
email: zyqiu@pku.edu.cn, {caic,zxp,yhl}@math.pku.edu.cn

Abstract

With the growth of interest on the web services, people have paid more attentions on the choreography, that is, to describe collaborations of participants from a global viewpoint, in accomplishing a common business goal. WS-CDL is the official language proposed by W3C for the specification of choreography. In this paper, based on a simple choreography language and a role-oriented process language, we studied some fundamental issues of choreography, and discussed a number of basic concepts, including semantics, projection and natural projection, well-formedness and restricted natural choreography, implementation and conformance problems, etc. The results of this study can help us to have a deeper understanding of choreography, and to have a better understanding of the problems related to the language design, implementation, wellformedness checking etc., and the specification and verification of choreographies.

1 Introduction

Web services have been becoming more important recently, which promise the interoperability of various applications running on heterogeneous platforms over Internet. Web service composition refers to the process of combining web services to provide value-added services, which has received much interest to support enterprise application integration. WS-CDL (for Web Service Choreography Description Language) [8, 7] is a W3C candidate recommendation for web service composition designed for describing the common and collaborative observable behavior of multiple services that interact with each other to achieve a business goal. WS-CDL focuses on specifying the business protocol among participants (roles). All the behaviors are performed by the participants, and the WS-CDL specification gives a global observation.

Voluminous literatures exist on the specification of systems, for example, [6] and [10]. However, almost all of them discussed the specification from a local viewpoint. Even for the communication, the description is still local, as it expresses when a process sends or receives a message from a specific channel. Furthermore, there is not a special concept of the *participants* of the computation. By the blooming of web-technology, more and more real computation are established in a kind of processes that various computing facilities take part in. These facilities are independent entities and might reside in any place over

*Supported by NNSF of China (No. 605730081)

the world. For accomplishing the goal of the computation, they should not only have "correct" functionalities, but also correctly interact with each other. With the interaction becoming more complex, the problems related to specify the interaction of the participants will be harder too, if we still want to do it locally. Moreover, it is even harder to verify the interaction locally. These are the motivation under the design of WS-CDL. Thus, a deeper understanding of the choreography and various relative problems is very urgent and important for the successful development of the web-based computation and application systems. It is better done at a suitable abstract level, to make a clear scenery of choreography. This is the motivation of the work presented here.

For exploring the essence of choreography, we defined a small language *Chor*, a model of the simplified WS-CDL, and a simple process language for the description of roles from a local viewpoint, both with formal syntax and semantics. Based on these models, we discussed the concept of projections, which map a given choreography to a set of role processes. We proposed a special map named *natural projection*, which partitions effectively choreography by following its structure. With the semantics of the languages and the natural projection, we proposed another level of well-formedness, and defined the concept of *restricted natural choreography* based on them. We proposed two structural conditions as a criterion to distinguish the restricted natural choreography. Then we discussed other possibilities of projection, especially, a projection which can remedy the sequential order problem of choreographies which are not restricted natural.

Using a projection, we will get a set of processes, where each of them represents a role in the choreography. We studied the implementation of the roles, which is commonly called the *conformance problem*. What we concerned in this work is better named *local conformance*, because the implementation of each role is considered independently.

After the sections on the results of this study, we had a section to discuss some important problems related to choreography in detail, including a comparison between local and global conformance, rationality of the two semantics proposed, some extensions and criticism of the choreography description languages. In this study, we have also given a large number of small examples, to help us and the readers to get a better understanding of the phenomena appeared here, and various properties of choreography.

The rest of the paper is organized as follows. In Section 2 and Section 3, we define the language *Chor* and a simple process language. Section 4 devotes to an important concept, *projection* and various issues related to it. In Section, we consider another revised semantics and various problems related to it. A number of general issues related to is discussed in Section 7, then the conclusion.

2 The Choreography Language *Chor*

We suppose that there is a finite set of roles taking part in the choreography C, where each role is associated with a number of basic activities:

$$\mathcal{R}_C = \{R^1, \ldots R^n\}$$
$$locals(R^i) = \{\mathsf{a}^i_1, \ldots, \mathsf{a}^i_{n_i}\}$$

We will use meta-variable a^i to denote an arbitrary activity of R^i, use $a, a_1, \ldots$ for activities of any role, and $locals(C)$ for the set of all local activities in C.

Figure 1: Syntax of *Chor*

$A ::=$	BA	(basic activities)	$BA ::=$	skip	(no action)
$\mid$	$A;\ A$	(sequential)	$\mid$	a^i	(activity in role R^i)
$\mid$	$A \sqcap A$	(choice)	$\mid$	$i \xrightarrow{c} j$	(communication)
$\mid$	$A \,\|\, A$	(parallel)			

The communication from role R^i to R^j takes the form $i \xrightarrow{c} j$, where c is the mark of the communication, R^i and R^j the sender and receiver respectively. We will use $\mathsf{c}, \mathsf{c1}, \ldots$ for concrete communication marks in examples. We take no care about the messages transferred, nor the variables receiving them.

The syntax of basic choreography language *Chor* is defined in Figure 1. A choreography (similar to a "program") is simply an action A. The activity set of choreography C, $acts(C)$, is the set of all activities appearing in C. We call role R^i (role i) the *performer* of local activity a^i, and role i and j the performers of communication $i \xrightarrow{c} j$. The activity set of role R_i is defined as:

$$acts(R^i) = \{\alpha \mid R^i \text{ is a performer of } \alpha\}$$

Beside, we will use $comms(C)$ and $comms(R^i)$ to denote the set of all communication activities in C or performed by R^i respectively. We will use $\alpha, \beta, \ldots$, possibly with subscript, to denote arbitrary (local or communication) activity.

2.1 Semantics

We consider the meaning of a choreography in *Chor* as the set of all possible traces of its execution. A trace $\langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ is a sequence of activities, where n is its length, possibly 0 for the empty trace. We will use $t, t_1, \ldots$ to denote traces, and $T, T_1, \ldots$ to denote sets of traces. We will use the ordinary operators on the sequences: $\frown$ for concatenation, hd and tl for head and tail respectively. We lift the $\frown$ operator to the trace set on either or both sides

$$t \frown T \mathrel{\widehat=} \{t \frown t' \mid t' \in T\}$$
$$T \frown t \mathrel{\widehat=} \{t' \frown t \mid t' \in T\}$$
$$T_1 \frown T_2 \mathrel{\widehat=} \{t_1 \frown t_2 \mid t_1 \in T_1, t_2 \in T_2\}$$

For the definition of the semantics of parallel composition, we need a function $interleave(T_1, T_2)$ which interleaves each pair of traces from T_1 and T_2 respectively, gives the set of all result traces. The definition is routine and omitted here. We all know that $interleave(T_1, T_2) = interleave(T_2, T_1)$.

We will also use a filter operation on a trace (or a trace set). For a trace t (or a trace set T) and a set S, $t \downharpoonleft S$ (or $T \downharpoonleft S$) retains only elements of S in the trace(s), and keeps the order within the trace(s) unchanged.

The rules for the semantics of *Chor* are listed in Figure 2. Here we adopt the interleaving semantics for parallel composition. This will have real consequence in the follows. We will discuss the problem later.

Definition 1 (Traces of Choreography) *The trace set of choreography A is the set $[\![A]\!]$ defined by the rules in* Figure 2. □

Figure 2: Semantics of *Chor*

Basic:	$[\![\mathsf{skip}]\!] \mathrel{\widehat=} \{\langle\,\rangle\}$	$[\![a]\!] \mathrel{\widehat=} \{\langle a\rangle\}$	$[\![i \xrightarrow{c} j]\!] \mathrel{\widehat=} \{\langle c^{[i,j]}\rangle\}$
Sequential:	$[\![A_1;\ A_2]\!] \mathrel{\widehat=} [\![A_1]\!] \frown [\![A_2]\!]$		
Choice:	$[\![A_1 \sqcap A_2]\!] \mathrel{\widehat=} [\![A_1]\!] \cup [\![A_2]\!]$		
Parallel:	$[\![A_1 \,\|\, A_2]\!] \mathrel{\widehat=} \mathit{interleave}([\![A_1]\!], [\![A_2]\!])$		

Figure 3: Laws in *Chor*

$(A_1;\ A_2);\ A_3 = A_1;\ (A_2;\ A_3)$	(; assoc.)	$A_1 \sqcap A_2 = A_2 \sqcap A_1$	($\sqcap$ sym.)
$(A_1 \sqcap A_2) \sqcap A_3 = A_1 \sqcap (A_2 \sqcap A_3)$	($\sqcap$ assoc.)	$A_1 \,\|\, A_2 = A_2 \,\|\, A_1$	($\|$ sym.)
$(A_1 \,\|\, A_2) \,\|\, A_3 = A_1 \,\|\, (A_2 \,\|\, A_3)$	($\|$ assoc.)	$\mathsf{skip}; A = A; \mathsf{skip} = A$	(; unit)
$(A_1 \sqcap A_2);\ A = (A_1;\ A) \sqcap (A_2;\ A)$	(;-$\sqcap$ distr.1)	$\mathsf{skip} \,\|\, A = A$	($\|$ unit)
$A;\ (A_1 \sqcap A_2) = (A;\ A_1) \sqcap (A;\ A_2)$	(;-$\sqcap$ distr.2)	$A \sqcap A = A$	($\sqcap$ idem.)
$(A_1 \sqcap A_2) \,\|\, A = (A_1 \,\|\, A) \sqcap (A_2 \,\|\, A)$	($\|$-$\sqcap$ distr.)		

2.2 Properties and Examples

Many laws hold for choreographies in *Chor*. We list some of them in Figure 3, where = represents semantical equivalence. The proofs are straightforward and are omitted. We might find many other laws about this language.

In company with other laws, we can use the *unit* and *idempotent* laws to remove all unnecessary skip from a choreography, and replace $A \sqcap A$ to A, to get a simplified choreography without changing its semantics.

The choices can always be moved out by the distribution laws related to $\sqcap$. By repeated using of these laws, any choreography can be transformed into a structure of the form $A'_1 \sqcap \ldots \sqcap A'_m$, where no choice appears in each of these $A'_1, \ldots, A'_m$. Using these laws in another direction, we can transform a choreography to make every $\sqcap$ to a minimal scope. Thus we have the following definition.

Definition 2 (Choice Normal Form) *A choreography in Chor of the form $A'_1 \sqcap \ldots \sqcap A'_m$, where no choice appearing in any of the $A'_1, \ldots, A'_m$, is called in the* Distributed Choice Normal Form. *A choreography in which the scope of every $\sqcap$ can not be limited is called in the* Restricted Choice Normal Form.

We know that a choreography in *Chor* can be transformed into an equivalent *Distributed Choice Normal Form*, or *Restricted Choice Normal Form.*

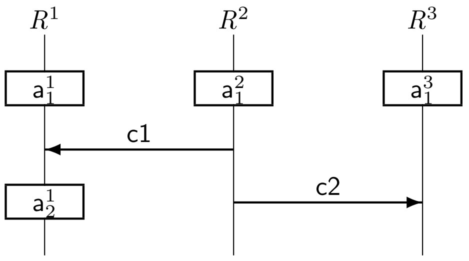
One interesting fact is, although a choreography in *Chor* may have parallel structures, no deadlock can happen in its execution.

Theorem 1 (Deadlock free) *Suppose that every basic activity in a choreography will terminate, then the choreography will always terminate.*

Proof. The theorem is obvious, because that all traces in the trace set of any choreography of *Chor* can be constructed. Even if we adopt the true concurrency semantics for the parallel structures, this theorem holds still. □

Now we give some examples, to illustrate characteristics of *Chor*.

Figure 4: A choreography which can not be described naturally in *Chor*



Exam. 1 *Here is a simple choreography written in Chor:*

$$C_1 \mathrel{\widehat{=}} (\mathsf{a}_1^1 \parallel \mathsf{a}_1^2);\ 1 \xrightarrow{\mathsf{c1}} 2;\ \mathsf{a}_2^2;\ 2 \xrightarrow{\mathsf{c2}} 1$$

Each of R^1 and R^2 performs a local activity in the first. Afterward, R^1 sends a message to R^2, then, R^2 performs a local activity and sends a message to R^1.

We can calculate the trace set of this choreography as follows:

$$\begin{aligned}[\![C_1]\!] &= [\![\mathsf{a}_1^1 \parallel \mathsf{a}_1^2]\!] \frown [\![1 \xrightarrow{\mathsf{c1}} 2]\!] \frown [\![\mathsf{a}_2^2]\!] \frown [\![2 \xrightarrow{\mathsf{c2}} 1]\!] \\ &= \{\langle \mathsf{a}_1^1, \mathsf{a}_1^2, \mathsf{c1}^{[1,2]}, \mathsf{a}_2^2, \mathsf{c2}^{[1,2]}\rangle, \langle \mathsf{a}_1^2, \mathsf{a}_1^1, \mathsf{c1}^{[1,2]}, \mathsf{a}_2^2, \mathsf{c2}^{[2,1]}\rangle\}\end{aligned}$$

This choreography has two possible traces. □

Exam. 2 *Here is another very simple choreography:*

$$C_2 \mathrel{\widehat{=}} (\mathsf{a}_1^1 \parallel \mathsf{a}_1^2);\ \mathsf{a}_2^1$$
$$[\![C_2]\!] = [\![\mathsf{a}_1^1 \parallel \mathsf{a}_1^2]\!] \frown [\![\mathsf{a}_2^1]\!] = \{\langle \mathsf{a}_1^1, \mathsf{a}_1^2, \mathsf{a}_2^1\rangle, \langle \mathsf{a}_1^2, \mathsf{a}_1^1, \mathsf{a}_2^1\rangle\}$$

This example will show some very interesting features of choreographies. We will return to this example and discuss it further in **Sec. 4.1**. □

Exam. 3 *Some very natural choreographies can not be described naturally in Chor (and also not in WS-CDL). We give a simple one in* Figure 4.

In the description, we should guarantee that a_1^3 can appear any time providing it occurs before $2 \xrightarrow{\mathsf{c2}} 3$, in the same time, a_2^1 can interleave with a_1^3 and $2 \xrightarrow{\mathsf{c2}} 3$ arbitrarily. It is not hard to see that the only means to describe it is to write the choreography as a choice, where each branch of it corresponds directly to a possible trace. This is really not ideal. The problem is not only the ugliness of the description. If we extend the language with structures producing infinite trace set (e.g., iteration or recursion), the similar cases will not be specifiable.

From our point of view, the incapability comes from the structural construction in the language, adopted by WS-CDL and also Chor. Although the structural composition is (almost) enough in sequential and even parallel programming, it is not clear whether these limited flow forms are enough in the choreography field. This should be investigated practically and theoretically. As an example, [5] proposes a graphic notation for choreography description. □

Figure 5: Syntax of the Role Language

P	$::=$	BP	(basic processes)	BP	$::=$	skip	(no action)
	$\mid$	$P;\ P$	(sequential)		$\mid$	a	(local activity)
	$\mid$	$P \sqcap P$	(choice)		$\mid$	$c!$	(send message)
	$\mid$	$P \parallel P$	(parallel)		$\mid$	$c?$	(receive message)

Figure 6: Semantics of the Role Language

Skip: $\mathsf{skip} \xrightarrow{\langle\,\rangle} \epsilon$ **Local:** $a \xrightarrow{\langle a \rangle} \epsilon$

Sequential: $\dfrac{P_1 \xrightarrow{\sigma} P_1'}{P_1;\ P_2 \xrightarrow{\sigma} P_1';\ P_2}$ $\epsilon;\ P_2 \xrightarrow{\langle\,\rangle} P_2$

Choice: $P_1 \sqcap P_2 \xrightarrow{\langle\,\rangle} P_1$ $P_1 \sqcap P_2 \xrightarrow{\langle\,\rangle} P_2$

Parallel: $\epsilon \parallel \epsilon \xrightarrow{\langle\,\rangle} \epsilon$ $\dfrac{P_1 \xrightarrow{\sigma} P_1'}{P_1 \parallel P_2 \xrightarrow{\sigma} P_1' \parallel P_2}$ $\dfrac{P_2 \xrightarrow{\sigma} P_2'}{P_1 \parallel P_2 \xrightarrow{\sigma} P_1 \parallel P_2'}$

$c!;\ P_1 \parallel c?;\ P_2 \xrightarrow{\langle c \rangle} P_1 \parallel P_2$ $c?;\ P_1 \parallel c!;\ P_2 \xrightarrow{\langle c \rangle} P_1 \parallel P_2$

3 A Process Language for the Roles

A choreography describes the interaction between some roles from a global view. It is intended to be implemented by the coordination of a set of independent processes. Now we study the relationship between the globally described choreography and the coordinative activities of each roles. In this section, we define a simple process language for the description of roles, and then, in the next section, we will we investigate the relationship mentioned above.

The process language used here is similar to the ordinary process algebra CSP. The syntax of the language is given in Figure 5. The only difference of this language from *Chor* is that it takes the local view about communications. There are send and receive activities, because they represent the role's local viewpoint about the communication. As in CSP, a send and a receive engage into a handshake when they have the same channel name (marked the same c) and the two roles involved are ready to perform them.

We can also define the sets $locals(P)$, $comms(P)$ and $acts(P)$. The definitions are similar to what in *Chor* (**Sec. 2**), and are omitted here. We use ϵ to denote the empty process, i.e., the process with no command.

We consider the semantics of a process in the role language as a set of traces too. An operational semantics to the language is defined by rules of the form $P \xrightarrow{\sigma} P'$, where P and P' are processes, and σ is a sequence of activities (a trace), to mean that after execution of the activities in σ from process P, we will arrive process P'. The semantical rules are given in Figure 6.

If in a situation, no rule defined can be applied to a process which is not empty, we say that the process is deadlocked, use a special symbol $\boxtimes$ to represent this situation, and introduce a rule:

$$\frac{P \neq \epsilon \text{ and } P \text{ is deadlocked}}{P \xrightarrow{\langle \boxtimes \rangle} \epsilon}$$

For the definition of the trace of the processes, we define:

$$\frac{P \xrightarrow{\sigma} P'}{P \overset{\sigma}{\Longrightarrow} P'} \qquad \frac{P \xrightarrow{\sigma} P' \quad P' \overset{\sigma'}{\Longrightarrow} P''}{P \overset{\sigma^\frown\sigma'}{\Longrightarrow} P''}$$

Now, we can have the definition of the trace set corresponding to a process.

Definition 3 (Traces of Process) *If $P \overset{\sigma}{\Longrightarrow} \epsilon$, then σ is called a trace of P. The trace set of process P is defined as the semantics of P:*

$$[\![P]\!] = \{\sigma \mid P \overset{\sigma}{\Longrightarrow} \epsilon\}$$

Although we adopt the same notation $[\![\cdot]\!]$ for the semantics of the two different languages, it will not cause confusion.

We can also explore the equivalent relation between processes. All the laws presented in **Sec. 2.2** hold for the processes. Some of these laws will be used in **Sec. 4** etc., in the simplification of the processes produced by the projections. People have proposed many more useful laws for other similar process language, which are out the scope of this study.

The choreography given in **Exam. 3**, which does not have a natural description in *Chor*, is easy to defined using this process language naturally.

$$R^1 \mathrel{\widehat{=}} \mathsf{a}^1_1;\ \mathsf{c1?};\ \mathsf{a}^1_2 \qquad R^2 \mathrel{\widehat{=}} \mathsf{a}^2_1;\ \mathsf{c1!};\ \mathsf{c2!} \qquad R^3 \mathrel{\widehat{=}} \mathsf{a}^3_1;\ \mathsf{c2?}$$

Now $R1 \parallel R2 \parallel R3$ has the intended behavior. This shows the different expressive power of these two languages.

4 Projection

A choreography is a global description of a task participated by a number of partners. With a choreography, we need a concept of its implementation. From our point of view, a reasonable definition of the implementation is based on the concepts of *projection* and *local conformance*. We will consider the issues related to projection in this section.

Some researchers proposed other definitions for the implementation of choreography, which can be named as *global conformance*, for example [3]. Briefly speaking, our definition pays more appreciation on the roles described in the choreography, and think them as concrete entities taking part in the interactions. We have some discussion and comparison in **Sec. 6** on the rationale of the two definitions.

By *projection*, we mean a procedure which takes a choreography in *Chor* and delivers a set of processes in the role language, while each of the processes corresponds to a role in the choreography. We wish that a projection can partition a choreography into a set of processes which can mimic the behavior described by the choreography, that is, for projection *proj*, we wish that:

$$[\![\mathit{proj}(C, 1) \parallel \ldots \parallel \mathit{proj}(C, n)]\!] = [\![C]\!] \qquad (1)$$

In this sense, we think that these processes make up an implementation of the choreography. Please note that, if the execution of the left hand side may engage

Figure 7: The Natural Projection

$nproj(\mathsf{skip}, k)$	$\widehat{=}$ skip			
$nproj(a^i, k)$	$\widehat{=}$ a^i	if $k = i$		
$nproj(a^i, k)$	$\widehat{=}$ skip	if $k \neq i$		$nproj(A_1;\ A_2, k)$ $\widehat{=}$ $nproj(A_1, k);\ nproj(A_2, k)$
$nproj(i \xrightarrow{c} j, k)$	$\widehat{=}$ $c^{[i,j]}!$	if $k = i$		$nproj(A_1 \sqcap A_2, k)$ $\widehat{=}$ $nproj(A_1, k) \sqcap nproj(A_2, k)$
$nproj(i \xrightarrow{c} j, k)$	$\widehat{=}$ $c^{[i,j]}?$	if $k = j$		$nproj(A_1 \parallel A_2, k)$ $\widehat{=}$ $nproj(A_1, k) \parallel nproj(A_2, k)$
$nproj(i \xrightarrow{c} j, k)$	$\widehat{=}$ skip	if $k \neq j$ $\wedge k \neq j$		

in a deadlock, some of its trace will end with the special symbol ⊠. No trace in $[\![C]\!]$ has this symbol as a part, thus, in this case, the equation will never hold.

There is not a standard definition on the projection. We will study a simplest one in **Sec. 4.1** with some discussion, and try to consider the problems recognized from this study afterward.

4.1 Natural Projection

It is natural that a projection works by directly following the structure of the choreography. The simplest projection we define is a simple partition procedure, named *natural projection.* The projection takes a choreography and a valid role number as arguments, produces a role description in our process language. Giving it all valid role numbers, we can get a set of processes, each for a role.

Definition 4 (Natural Projection) *The definition is in* Firgure 7*, where the basic cases are on the left, and the structural on the right. We think $[i, j]$ as a part of the channel name. The projection follows the structure of the choreography, and partitions it according to the role number.* □

The projection may leave many skip and $P \sqcap P$ in the role process. We can introduce a procedure *simp* to simplify the result and keep the semantics, using the rules in Figure 3. Because what we concern the most is the semantics, we will not care about the details of the simplification in the following, and simply write down the simplified version of the process.

Exam. 4 *Let us apply nproj to the choreography of* **Exam. 1***:*

$$nproj(C_1, 1) = \mathsf{a}_1^1;\ \mathsf{c1}^{[1,2]}!;\ \mathsf{c2}^{[2,1]}?$$
$$nproj(C_1, 2) = \mathsf{a}_1^2;\ \mathsf{c1}^{[1,2]}?;\ \mathsf{a}_2^2;\ \mathsf{c2}^{[2,1]}!$$

It is easy to see that $[\![nproj(C_1, 1) \parallel nproj(C_1, 2)]\!] = [\![C_1]\!]$. □

However, this is not always the case. Let us see another example.

Exam. 5 *Consider the choreography in* **Exam. 2**. *We have:*

$$nproj(C_2, 1) = \mathsf{a}_1^1;\ \mathsf{a}_2^1$$
$$nproj(C_2, 2) = \mathsf{a}_1^2$$

Thus $[\![nproj(C_2, 1) \parallel nproj(C_2, 2)]\!] = \{\langle \mathsf{a}_1^2, \mathsf{a}_1^1, \mathsf{a}_2^1\rangle, \langle \mathsf{a}_1^1, \mathsf{a}_1^2, \mathsf{a}_2^1\rangle, \langle \mathsf{a}_1^1, \mathsf{a}_2^1, \mathsf{a}_1^2\rangle\}$. Obviously, it is not the same as $[\![C_2]\!]$ (see **Exam. 2**). □

From this example, we know that the natural projection do not guarantee that the parallel composition of the partitioning results has the same behavior as the original choreography. When (1) does not hold, we could impute the guilt to the choreography under consideration, or as well as, to the projection defined. We will investigate the problem further following these two points of view.

4.2 Restricted Natural Choreographies

It seems that the *natural projection* is very intuitive. Thus, from one point of view, we can take this projection as a criterion to distinguish the "good" choreographies from the "bad" ones, and might say that the choreography in **Exam. 1** is "good', but the one in **Exam. 2** is "bad". From this view point, we have the following definition:

Definition 5 (Restricted Natural Choreography) *Suppose C is a choreography with n roles. We call C a* restricted natural choreography*, if*

$$[\![nproj(C,1) \parallel \ldots \parallel nproj(C,n)]\!] = [\![C]\!] \qquad (2)$$

We use the name of *restricted natural choreography* (*RN* choreography), because we think that this class of choreography is natural, but also too restricted (as we will see in the following). We hope that by some approach similarly, we can find a useful class of *natural choreography*, which can catch the concept "implementable" reasonably and effectively.

In the rest part of this section, we will take the RN choreographies as "good" ones, i.e., think that **Definition 5** defines a restricted level of well-formedness. If a choreography is RN, we can efficiently partition it into a set of independent processes, and these processes will show the expected behavior.

Although the definition above is accurate, it is not easy to use. Now we will look for the structural features of RN choreographies. We want to answer the problem: what structures make an RN choreography? We consider some sufficient conditions here. In the rest of this subsection, when we use the word "projection", we always mean the natural projection.

4.2.1 Sequential Composition

The problem related to sequential composition is the keeping of the order in the result processes produced by the projection. We have seen a counterexample in **Exam. 5**. Because the processes produced by natural projection can not keep the relative order of their activities, thus, an extra trace appears.

We need to ask the projection keeping the original relative order between the different roles. For description of the structural condition on the sequential composition, we define two activity sets in the first:

Definition 6 (Leading and Ending Activity Sets) *For any activity (either* *basic or not) we define two activity sets recursively as follows:*

$$\begin{aligned}
&\mathit{lead}(\mathsf{skip}) = \mathit{end}(\mathsf{skip}) = \varnothing\\
&\mathit{lead}(\alpha) = \mathit{end}(\alpha) = \{\alpha\}\\
&\mathit{lead}(A_1 \parallel A_2) = \mathit{lead}(A_1) \cup \mathit{lead}(A_2)\\
&\mathit{lead}(A_1 \sqcap A_2) = \mathit{lead}(A_1) \cup \mathit{lead}(A_2)\\
&\mathit{lead}(A_1;\ A_2) = \begin{cases} \mathit{lead}(A_2) & \text{if } A_1 \text{ equivalent to } \mathsf{skip}\\ \mathit{lead}(A_1) & \text{otherwise}\end{cases}\\
&\mathit{end}(A_1 \parallel A_2) = \mathit{end}(A_1) \cup \mathit{end}(A_2)\\
&\mathit{end}(A_1 \sqcap A_2) = \mathit{end}(A_1) \cup \mathit{end}(A_2)\\
&\mathit{end}(A_1;\ A_2) = \begin{cases} \mathit{end}(A_1) & \text{if } A_2 \text{ equivalent to } \mathsf{skip}\\ \mathit{end}(A_2) & \text{otherwise}\end{cases}
\end{aligned}$$

In words, lead(A) includes all activities which can be executed at the beginning of the execution of A, and end(A) includes all activities which can be the last activity of the execution of A. □

Now we can write down the condition where each sequential composition in a natural choreography must satisfy:

Condition 1 (Sequential Composition) *A sequential composition $A_1;\ A_2$ is restricted natural (is RN), if it satisfy:*

$$\forall\, \alpha_1 \in \mathit{end}(A_1), \alpha_2 \in \mathit{lead}(A_2) \bullet \alpha_1 \textit{ and } \alpha_2 \textit{ have a common performer.} \qquad (3)$$

The concept performer *is defined in* **Sec. 2**. □

Here are some choreographies where all sequential compositions are RN:

$$\begin{aligned}
&\mathsf{a}^1_1;\ \mathsf{a}^1_2\\
&(\mathsf{a}^1_1 \parallel \mathsf{a}^2_1);\ 1 \xrightarrow{\mathsf{c}} 2\\
&(1 \xrightarrow{\mathsf{c1}} 3 \parallel (\mathsf{a}^2_1;\ 2 \xrightarrow{\mathsf{c2}} 4));\ 1 \xrightarrow{\mathsf{c3}} 2;\ \mathsf{a}^1_1
\end{aligned}$$

The last example shows, when one side of a sequential composition is a parallel, we need to consider each of its branches separately. It is also the case for choices. Because choices admit more restrictions, we omit examples involving it here, and will discuss its details in the next subsection.

Exam. 6 *A simplest counterexample is $\mathsf{a}^1_1;\ \mathsf{a}^2_1$. After the projection we get*

$$\begin{aligned}
&P_1 = \mathsf{a}^1_1\\
&P_2 = \mathsf{a}^2_1
\end{aligned}$$

In execution of $P_1 \parallel P_2$, the relative order between a^1_1 and a^2_1 can not be kept. The trace set becomes $\{\langle \mathsf{a}^1_1, \mathsf{a}^2_1\rangle, \langle \mathsf{a}^2_1, \mathsf{a}^1_1\rangle\}$. If we add a communication to separate these two activities to get $\mathsf{a}^1_1;\ 1 \xrightarrow{c} 2;\ \mathsf{a}^2_1$, this new choreography is RN, because the synchronization between these two processes enforces the order. The choreography in **Exam. 2** *can be amended into an RN one in the similar way too.*

Exam. 7 *Condition* (3) *has a sad consequence. In a choreography, we have no way to enforce local activities of more than two roles to terminate at the same time, when they run in different parallel branches. Here is an illuminating example. Consider a simple non-RN choreography* $C_4 \mathrel{\widehat{=}} (\mathsf{a}^1_1 \parallel \mathsf{a}^2_1 \parallel \mathsf{a}^3_1);\ \mathsf{a}^1_2$*. We can not even make an RN choreography based on it by adding some synchronization activities. Suppose we add two communications as follows*

$$C'_4 \mathrel{\widehat{=}} (\mathsf{a}^1_1 \parallel \mathsf{a}^2_1 \parallel \mathsf{a}^3_1);\ 1 \xrightarrow{\mathsf{c1}} 2;\ 1 \xrightarrow{\mathsf{c2}} 3;\ \mathsf{a}^1_2$$

Then we will have:

$$\begin{aligned}
&\mathit{nproj}(C'_4, 1) = \mathsf{a}^1_1;\ \mathsf{c1}^{[1,2]}!;\ \mathsf{c2}^{[1,3]}!;\ \mathsf{a}^1_2\\
&\mathit{nproj}(C'_4, 2) = \mathsf{a}^2_1;\ \mathsf{c1}^{[1,2]}?;\\
&\mathit{nproj}(C'_4, 3) = \mathsf{a}^3_1;\ \mathsf{c2}^{[1,3]}?;\\
&[\![C'_4]\!] = \{\langle \mathsf{a}^1_1, \mathsf{a}^2_1, \mathsf{a}^3_1, \mathsf{c1}^{[1,2]}, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle, \langle \mathsf{a}^1_1, \mathsf{a}^3_1, \mathsf{a}^2_1, \mathsf{c1}^{[1,2]}, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle,\\
&\qquad\quad \langle \mathsf{a}^2_1, \mathsf{a}^1_1, \mathsf{a}^3_1, \mathsf{c1}^{[1,2]}, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle, \langle \mathsf{a}^2_1, \mathsf{a}^3_1, \mathsf{a}^1_1, \mathsf{c1}^{[1,2]}, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle,\\
&\qquad\quad \langle \mathsf{a}^3_1, \mathsf{a}^1_1, \mathsf{a}^2_1, \mathsf{c1}^{[1,2]}, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle, \langle \mathsf{a}^3_1, \mathsf{a}^2_1, \mathsf{a}^1_1, \mathsf{c1}^{[1,2]}, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle, \}\\
&[\![\mathit{nproj}(C'_4, 1) \parallel \mathit{nproj}(C'_4, 2) \parallel \mathit{nproj}(C'_4, 3)]\!] =\\
&\qquad [\![C'_4]\!] \cup \{\langle \mathsf{a}^1_1, \mathsf{a}^2_1, \mathsf{c1}^{[1,2]}, \mathsf{a}^3_1, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle, \langle \mathsf{a}^2_1, \mathsf{a}^1_1, \mathsf{c1}^{[1,2]}, \mathsf{a}^3_1, \mathsf{c2}^{[1,3]}, \mathsf{a}^1_2\rangle\}
\end{aligned}$$

Other arrangements will have the similar results. □

Some researchers proposed to add a kind of multipartite communication activity to remend this problem, for example [2].

4.2.2 Choice

The natural projection maps a choice in a choreography to a choice in each of the role processes. After parallel composition, these role processes will run independently. Thus, we have no way to guarantee that they will take consistently the same branch when they run to their own version of the choice. The inconsistent choice can result extra traces, or even result deadlocked traces. We will have some examples below to illustrate various situations.

The simplest example showing the problem clear, is choreography $\mathsf{a}^1_1 \sqcap \mathsf{a}^2_1$. It has trace set $\{\langle \mathsf{a}^1_1\rangle, \langle \mathsf{a}^2_1\rangle\}$. After the projection and parallel composition, we have $(\mathsf{a}^1_1 \sqcap \mathsf{skip}) \parallel (\mathsf{a}^2_1 \sqcap \mathsf{skip})$ with trace set $\{\langle\,\rangle, \langle \mathsf{a}^1_1\rangle, \langle \mathsf{a}^2_1\rangle, \langle \mathsf{a}^1_1, \mathsf{a}^2_1\rangle, \langle \mathsf{a}^2_1, \mathsf{a}^1_1\rangle\}$.

For a choice to bring no trouble, one possibility is that only one real choice appears in all role processes after the projection. Thus we have a condition:

Condition 2 (Choice) *For a choice, if we can determine a dominant role from it, where for all the other roles involved, the activities related to each of them are consistent in all branches of the choice, then the choice is an RN choice. An activity is* related to *a role if the role is its performer.* □

This condition means that, after projection of an RN choice, each role processes of A produced by the projection can reduce to a process without choice, using the laws (Figure 3) related to "$\sqcap$", except that at most one of them is not the case, which is the process corresponding to the *dominant role* of the choice structure, and it can do anything.

Now we give some examples to illustrate the condition. In the following example, we assume the implicit simplification is extended with laws related to choice, to merge identical choice branches.

Exam. 8 *Choreography C_5 includes an RN choice with communication activities:*

$$C_5 = \mathsf{a}_1^1;\ ((\mathsf{a}_2^1;\ 1\xrightarrow{\mathsf{c}}2) \sqcap (\mathsf{a}_3^1;\ 1\xrightarrow{\mathsf{c}}2));\ \mathsf{a}_1^2$$

R^1 is the dominant role in this choice. The projection produces

$$\begin{aligned}&nproj(C_5, 1) = \mathsf{a}_1^1;\ (\mathsf{a}_2^1;\ \sqcap(\mathsf{a}_3^1));\ \mathsf{c}^{[1,2]}!\\ &nproj(C_5, 2) = \mathsf{c}^{[1,2]}?;\ \mathsf{a}_1^2\end{aligned}$$

It is easy to verify that $[\![C_5]\!] = [\![nproj(C_5, 1) \parallel nproj(C_5, 2)]\!]$. □

Exam. 9 *Here is a negative example with communication activities:*

$$\begin{aligned}&C_6 = \mathsf{a}_1^1;\ ((\mathsf{a}_2^1;\ 1\xrightarrow{\mathsf{c1}}2) \sqcap (\mathsf{a}_3^1;\ 1\xrightarrow{\mathsf{c2}}3));\ \mathsf{a}_1^2\\ &nproj(C_6, 1) = \mathsf{a}_1^1;\ ((\mathsf{a}_2^1;\ \mathsf{c1}^{[1,2]}!) \sqcap (\mathsf{a}_3^1;\ \mathsf{c2}^{[1,3]}!))\\ &nproj(C_6, 2) = (\mathsf{c1}^{[1,2]}? \sqcap \mathsf{skip});\ \mathsf{a}_1^2\\ &nproj(C_6, 3) = (\mathsf{c2}^{[1,3]}? \sqcap \mathsf{skip})\end{aligned}$$

Three processes produced by the projection can not have a consistent view on their choices. The parallel composition of them may run into deadlock. □

4.2.3 Structural Theory of RN Choreography

For the parallel composition, we find no extra conditions to make the choreography RN. The interaction between parallel and sequential composition has been studied in **Sec. 4.2.1**. Now we think about the interaction between the parallel composition and the choice. **Exam. 8** is an example for this interaction. We give some more illustrative examples here:

Exam. 10 *Consider $C_7 = ((\mathsf{a}_1^1;\ 1\xrightarrow{\mathsf{c1}}2) \sqcap (\mathsf{a}_2^1;\ 1\xrightarrow{\mathsf{c1}}2)) \parallel (2\xrightarrow{\mathsf{c2}}1;\ \mathsf{a}_1^2)$. The dominant role in the choice is R^1. By the projection we have:*

$$\begin{aligned}&P_1 = ((\mathsf{a}_1^1;\ \mathsf{c1}^{[1,2]}!) \sqcap (\mathsf{a}_2^1;\ \mathsf{c1}^{[1,2]}!)) \parallel \mathsf{c2}^{[2,1]}?\\ &P_2 = \mathsf{c1}^{[1,2]}? \parallel (\mathsf{c2}^{[2,1]}!;\ \mathsf{a}_1^2)\end{aligned}$$

No matter which branch P_1 chooses, it can synchronize with P_2. It is not hard to see that we have $[\![C_7]\!] = [\![P_1 \parallel P_2]\!]$.

Now consider an example similar to the one above, but involving parallel composition of two choices with different dominant roles:

$$C_8 = ((\mathsf{a}_1^1;\ 1\xrightarrow{\mathsf{c1}}2) \sqcap (\mathsf{a}_2^1;\ 1\xrightarrow{\mathsf{c1}}2)) \parallel ((2\xrightarrow{\mathsf{c2}}1;\ \mathsf{a}_1^2) \sqcap (\mathsf{a}_2^2;\ 2\xrightarrow{\mathsf{c2}}1))$$

The dominant roles of two choices are R^1 and R^2, respectively. We have:

$$\begin{aligned}&P_1 = ((\mathsf{a}_1^1;\ \mathsf{c1}^{[1,2]}!) \sqcap (\mathsf{a}_2^1;\ \mathsf{c1}^{[1,2]}!)) \parallel \mathsf{c2}^{[2,1]}?\\ &P_2 = \mathsf{c1}^{[1,2]}? \parallel ((\mathsf{c2}^{[2,1]}!;\ \mathsf{a}_1^2) \sqcap (\mathsf{a}_2^2;\ \mathsf{c2}^{[2,1]}!))\end{aligned}$$

No matter which branches P_1 and P_2 choose, the synchronization will not meet a problem. We can easily determine that $[\![C_8]\!] = [\![P_1 \parallel P_2]\!]$. □

Ideally, we hope to find a set of structural conditions, which are able to distinguish the RN chorographies from the others, both necessary and sufficient. However, we can only prove that the conditions given above are sufficient.

Theorem 2 (Restricted Natural Choreography) *Suppose choreography C is in its* Minimal Choice Normal Form (**Definition 2**)*, if each of the sequential composition and choice in C satisfies* **Condition 1** *or* **Condition 2** *respectively, then C is restricted natural.*

Proof. The proof is lengthy but not hard. We put it in the ***Appendix***. □

A simple counter example that shows the conditions are not necessary is $(\mathsf{a}_1^1;\ \mathsf{a}_1^2) \sqcap (\mathsf{a}_1^2;\ \mathsf{a}_2^1)$. It is restricted natural but violates **Condition 1**.

4.3 Revising the Projection

As pointed in **Sec. 4.1**, we can also blame the projection defined (the *nproj*) for the failure of Equation (1) in general. Taking other definitions may make an equation similar to (1) hold for more choreographies. There are many possibilities to define such projections. We consider one of them in this subsection.

Now we want to remedy the ordering problem in the partitioning of the roles. From the discussion and the examples above, the approach is quite clear. To enforce the order of activities between different role processes, we can insert extra communication activities in the proper positions to synchronize these processes. We are going to define a procedure, given a choreography C with n roles, which might violate **Condition 1** (but not violate **Condition 2**), the procedure inserts some communication activities (in either direction) in some positions of C, to produce a revised choreography C', such that we can ensure

$$[\![\mathit{nproj}(C',1) \parallel \ldots \parallel \mathit{nproj}(C',n)]\!] \downharpoonright \mathit{acts}(C) = [\![C]\!] \tag{4}$$

here $\downharpoonright$ is the filter operation described in **Sec. 2.1**.

The insertion procedure is clear: We need to check each sequential composition in the choreography, to see if the insertion is necessary. In general, for a composition $A_1;\ A_2$, suppose $S_1 = \mathit{end}(A_1)$ and $S_2 = \mathit{lead}(A_2)$. For each activity $a \in S_2$, we insert a sequence M_a of communication activities as the sequential precedents of a, where the communications in M_a "links" the performer of a with all the roles which are the performers of activities in S_1.

The general philosophy is, with the enough insertions of communication, we can separate the activities of different roles in some steps, force the roles related synchronizing with each other in these position. Of course, there might be some optimization. For example, we should not use communication to synchronize activities of the same role, etc. There are some other special cases that may be used to reduce the number of insertions:

- If either S_1 or S_2 includes only one activity, which covers at least one role involved in each activity on the other hand, then we should not insert anything between A_1 and A_2.

- If there are communication activities in S_1 and/or S_2, the separating communication set need links only one performer for each communication.

The insertion procedure is easy to define rigorously. We omit the details here and rely on the readers' intuition. We give in the follows some examples to illustrate the procedure, and show some special situations.

Exam. 11 *For the simplest case $C = a^i;\ a^j$, $(i \neq j)$, we insert a communication from R^i to R^j, to get $C' = a^i;\ i \xrightarrow{c} j;\ a^j$. It satisfies* (4).

For choreography $(a^1 \parallel 2 \xrightarrow{c1} 3);\ 1 \xrightarrow{c2} 2$, we don't need to insert anything.

Now consider C_4 in **Exam. 7***, which involves parallel composition of local activities of three roles. Inserting two communications to get C_4', we have*

$$[\![\mathit{nproj}(C_4', 1) \parallel \mathit{nproj}(C_4', 2) \parallel \mathit{nproj}(C_4', 3)]\!] \downharpoonleft \mathit{acts}(C_4) = [\![C_4]\!]$$

Now Equation (4) *holds, because the additional activities are filtered out. Comparing this with* **Exam. 7***, the equation considered is not the same.* □

From these examples, we know that this projection with Equation (4) can admit some choreographies which are not RN. We can follow the approach used in **Sec. 4.2**, and define an extended class of "good" choreography.

The problem related to **Condition 2** can not be solved in current framework, because it involves the consistent choice in different roles. To solve this problem, we need not only the communication ability, but also a mechanism to make choice according to the communication. We can extend the role language with the guarded command and external choice structure, and then try to solve this problem. This possibility will be an interesting future work.

5 Conformance and Implementation

With a projection defined, we can get a set of processes from a choreography which represent all the roles taking part in the task described by the choreography. In this section, we will turn to define the concept of the implementation of a role. Naturally, if a process can play as a role in a choreography, it can take part in the choreography, and play with the other valid roles.

We can think that the roles of a choreography define a set of requirements on a set of concrete processes (or, web services). In this case, when having a process at hand, we should have a way to decide if this process can play as a specific role. This is called the *conformance* problem. It want to determine if a process conforms with a specific requirement expressed by a role. As said before, what we define and discuss here can be named *local conformance*, because it want to determine locally a relation between a role (requirement) and a process (a potential implementation).

In fact, there can be different definitions on the conformance, with less or more restricted conditions. We will discuss some possibilities in this section. We want to have an informal discussion about the rationality of the conformance in the first, and then the formal treatments.

5.1 Basic Consideration

Because we think a choreography as a behavior and interaction protocol of a specific task performed by a number of roles, any process which can join the interplay as a role must obey the specific part of the protocol related to the role. In the first, we have some basic assumptions about a choreography:

- It describes all important roles token part in the task. Each role should be implemented by an distinguishable independent process (an independent web-service) in the implementation. This also means that there should not be other substantial roles taking part in the interaction, except for the implementation of some local activities for some role(s).

- It describes all important message passing between the roles. The implementation should not add in other substantial interactions between roles, that is, no additional (substantial) message passing is allowed.

- The local activities described for each role are important. The implementation of a role (a process or a web-service) must perform the local activities of the specific role in a distinguishable way.

In fact, these assumptions are the fundamentals of our study on the projection and conformance. It has deep effect on our discussions in the last section (**Sec. 4**). In a special case, the extended projection proposed in **Sec. 4.3** does insert additional communications into the "implementation". We think that they are not substantial, because the only effect of them is synchronization, and no real message is passed in these communications.

When we have a role (protocol) R in a choreography C, a process P can be think as an implementation of R, written $P \trianglerighteq R$, if

- P can execute all communication required by R with other roles of C, in suitable time, with suitable order.

- P support all the local activities mentioned in R.

We can allow P to support extra functions, or even to delegate part of its local work to some other processes (web services), provided that those processes do not communicate with other roles in choreography C during their work. On the other word, it is forbidden that these delegations form new communication pathes between the roles other than what described in C.

Practically, this kind of delegation seems unavoidable. For example, a seller service often needs to ask a relative storage management service to check if there are enough amount of required goods. By taking different allowances, we can have different definitions of the conformance.

5.2 Play

For the formal definition of the conformance, we use the concept of *play*: a process is an implementation of a role, if the process can play the role. There are possibly different definitions for the play, we will define, initially, the most general one, and then some special and most useful ones related to the conformance problems concerned here.

In the following definitions, we use meta variable $\omega, \omega_1, \ldots$ to represent sequences of activities. Now, the general definition.

Definition 7 (General Play) *A process P can play a role R of choreography C, if and only if a binary relation $\trianglerighteq$ exists between them, that satisfies:*

(1) $\epsilon \trianglerighteq \epsilon$.

(2) If $R \xrightarrow{\langle\alpha\rangle} R'$, then there exists two activity sequences ω_1, ω_2 (possibly empty) and a process P', such that $P \overset{\langle\omega_1,\alpha,\omega_2\rangle}{\Longrightarrow} P'$ and $P' \trianglerighteq R'$.

where $P \trianglerighteq R$ can also be written as $R \trianglelefteq P$. □

Definition 7 guarantees that P executes every activity of R in a distinguished way, with also other activities admitted. For expressing this clearly, we define the concept of *role traces*, which concerns activities performed by a role from a local view.

Definition 8 (Role Traces) *The role trace set of a role process is defined by the following rules:*

$$
\begin{aligned}
&[\![a]\!] = \{\langle a\rangle\}\\
&[\![c!]\!] = \{\langle c!\rangle\}\\
&[\![c?]\!] = \{\langle c?\rangle\}\\
&[\![P_1;\ P_2]\!] = [\![P_1]\!] \frown [\![P_2]\!]\\
&[\![P_1 \sqcap P_2]\!] = [\![P_1]\!] \frown [\![P_2]\!]\\
&[\![P_1 \parallel P_2]\!] = interleave([\![P_1]\!], [\![P_2]\!])
\end{aligned}
$$

Here we omit the synchronization problem. □

Now we have the following result:

Theorem 3 *If $P \trianglerighteq R$, then for each trace $t \in [\![R]\!]$, there is a trace $t' \in [\![P]\!]$ such that t is a sub-trace of t', written $t \preceq t'$, where $\preceq$ is defined recursively:*

$$
\begin{aligned}
&\langle\,\rangle \preceq \langle\omega\rangle\\
&\langle\alpha,\omega_1\rangle \preceq \langle\alpha,\omega_2\rangle \Leftrightarrow \langle\omega_1\rangle \preceq \langle\omega_2\rangle\\
&\langle\alpha,\omega_1\rangle \preceq \langle\beta,\omega_2\rangle \Leftrightarrow \langle\alpha,\omega_1\rangle \preceq \langle\omega_2\rangle \quad \text{when } \alpha \neq \beta
\end{aligned}
$$

Proof. Obviously from the definition of **Definition 7** and **Definition 8**. □

In fact, the **Definition 7** is too general to be accepted, because the ω's might include any activity, thus, generally, the result may possibly violate our basic assumptions proposed in the last subsection. For the more reasonable conformance, we define here some more restricted simulations.

Definition 9 (Play) *If $P \trianglerighteq R$ for a process P and a role R of choreography C, we have the following additional definitions:*

- *If all ω's introduced in* **Definition** 7 (2) *are empty, we say that P plays R exactly, and write $P \trianglerighteq_{exact} R$.*
- *If all ω's introduced in* **Definition** 7 (2) *include only local activity which do not belong to $acts(C)$, we say that P plays R with extra local activities, and write $P \trianglerighteq_a R$.*
- *If all ω's introduced in* **Definition** 7 (2) *include only communication activity with communication partners out of $\mathcal{R}_C$ (the role set of C), we say that P plays R with extra communication, and write $P \trianglerighteq_c R$.*

- *If all ω's introduced in* **Definition** 7 (2) *include only local activity which do not belong to $acts(C)$ or communication activity with communication partners out of $\mathcal{R}_C$, we say that P plays R with extra local activities and communication, and write $P \trianglerighteq_{ac} R$.* □

Definition 10 (Conformance and Implementation) *We have corresponding definitions for the conformance and implementation:*

- *If $P \trianglerighteq_{exact} R$, we say that P conforms to R exactly, or P is an exact implementation of R.*
- *If $P \trianglerighteq_{a} R$, we say that P conforms to R with relaxed local activity, or P is an implementation of R with relaxed local activity.*
- *If $P \trianglerighteq_{c} R$, we say that P conforms to R with with relaxed communication, or P is an implementation of R with with relaxed communication.*
- *If $P \trianglerighteq_{ac} R$, we say that P conforms to R with with relaxed local activity and communication, or P is an implementation of R with with relaxed local activity and communication.* □

6 Discussion

Semantics. The semantics used in this paper is not the only one reasonable, and we can think about it. In the semantics defined in Sec. 2, we admit that all the relative order between activities described in the choreography are important, even if they appear in different (and independent) roles. Thus, we distinguish the choreographies such as:

$$a^1;\ a^2 \qquad\qquad a^2;\ a^1 \qquad\qquad a^1 \parallel a^2$$

However, is the distinguishing of these choreographies important? Some other papers also keep the relative speed of different roles, but without a discussion about its rationality. We think that more practice and theoretical study is necessary before a conclusion.

Although the semantics used here is useful in exploring properties and problems of the choreography. We can also think about other definition of semantics. Now we are working on another reasonable semantics, called *Role-Oriented Semantics*, where the relative speed of different roles is not important, except when they synchronize with each other in the communication. We find that many results we had in this paper can be transferred to that semantics, and we can also get a class of choreography, defined by *natural projection.* These primitive results tell us that the concepts and the approach used in this paper are reasonable. It is too early to say which semantics is superior. Now we are trying to understand them in details.

Conformance. As written in the WS-CDL Recommendation [8], a choreography describes a multi-participant contract from a global perspective, as a replacement to natural languages which are used today for this purpose. The document does not define clearly and accurately what is a "correct" implementation of a choreography, but proposes a concept called *conformance.* It says that

"Each participant can then use the global definition to build and test solutions that conform to it. The global specification is in turn realized by combination of the local systems, on the basis of appropriate infrastructure support". This leaves the definition of implementation open, and makes some confusions.

From our recognition, there exist at least two basic viewpoints to define the implementation of a choreography: With the *Global View*, a choreography describes a (business) process. An implementation is a process which has the behavior represented by the choreography. N. Busi *et al.* [4] gave a definition following this vein. On the other hand, with the *Role-based View* or *Local View*, a choreography describes a (business) process implemented by a set of participants. An implementation is a set of processes, each of these process implements one of the roles in a clear way, and the combination of these processes implement the behavior represented by the choreography.

The fundamental dissimilitude of these two viewpoints is whether the roles and activities described in the choreography must have their reincarnations in an implementation. The local view says that it is really important, but the global view says it is not the case.

We think that the definition of implementation following the the global view is too loose. The extreme case is that an implementation with only one process, which executes all the activities of all roles of the choreography, and makes all the communication activities as local assignments. Our approach takes the way of projection and local conformance. Following this approach, each role will be a real process in any valid implementation which runs parallel with other processes corresponding to other roles. In this case, the projection and local conformance discussed in this paper play important roles in the implementation, and should be studied in details further.

Language. As shown by many evidences in this paper, we are still in the first step in defining a language for the choreography. One problem is the control flow structures, as we have a glimpse of it in **Exam. 3**. Another example is the controversial issues about the control structure *workunit* taken in the WS-CDL [2]. Aalst *et al.* listed some challenges including defining a "real" choreography language in their paper [12].

7 Conclusion

By the blooming of web-technology, more and more computation are established by processes (services) residing over the Internet. Due to the nature of web services interaction, to guarantee the correct interaction of independent, communicating services becomes even more critical [11]. This is the motivation under WS-CDL [8], which is designed for specifying choreography, i.e., the global observation of business protocol among participants (roles). A deeper understanding of the choreography and relative problems is urgent and important.

N. Busi *et al.* formalize choreography and orchestration with process algebras and conformance takes the form of a bisimulation-like relation [4]. J. Mendling and M. Hafner proposes a simple projection mechanism from WS-CDL to WS-BPEL [9]. Based on π-calculus, Decker *et al.* [5] provide an execution semantics for a language *Let's Dance*, which can describe the choreography with graphical notations. Another interesting reference is the Pi4SOA project [1]. However, none of them takes the idea we use here: to use the projection as a mean to define the well-formedness and implementable of the choreography.

In this paper, we explored some basic concepts related to the choreography. We defined a small language *Chor* with formal semantics, which is a simplified formal model of WS-CDL. We defined also a simple process language for the description of roles in a local viewpoint, with its syntax and semantics.

Based on these formal models, we discussed the procedure of projection, which maps a given choreography into a set of role processes. A special map named *natural projection* is discussed, which can effectively partition choreographies following the structures. In a previous work [13], this projection is used in the verification of the choreography. With the semantics and the natural projection, we defined the concept of *Restricted Natural Choreography* as another level of well-formedness. For checking whether a choreography is restricted natural (RN), we proposed two structural conditions as criterion to distinguish the RN choreography, and proved that they are sufficient in the **Appendix**. We discussed other projections, especially, a projection which can remedy the sequential order problem of choreographies which are not RN.

By applying a projection, we get a set of processes where each of them represents a role in the choreography. We studied the implementation of roles, which is the *conformance problem*. What we concerned here is better named *local conformance*, because the implementation of each role is considered independently. We had a discussion about the local conformance vs. the global one.

We also discussed some problems related to choreography, including a comparison between local and global conformance, rationality of the semantics, extensions of the choreography description languages, etc. In this study, we have also given many examples, to help the readers getting better understanding of the phenomena appeared here, and of various properties of choreography.

References

[1] Pi4soa. http://www.pi4soa.org/.

[2] A. Barros, M. Dumas, and P. Oaks. A Critical Overview of the Web Services Choreography Description Language. www.bptrends.com, 2005.

[3] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro. Choreography and orchestration: a synergic approach for system design. In *Proc. of ICSOC 2005, LNCS 3826*. Springer, 2005.

[4] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro. Choreography and orchestration conformance for system design. In *Proc. of COORDINATION 2006, LNCS 4038*, 2006.

[5] G. Decker, J.M. Zaha, and M. Dumas. Execution semantics for service choreographies. In *Proc. of WS-FM 2006, LNCS 4184*. Springer, 2006.

[6] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[7] N. Kavantzas. Aggregating web services: Choreography and WS-CDL. Technical report, Oracle Coporation, 2004.

[8] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web Services Choreography Description Language Version 1.0. http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/, Nov. 9, 2005.

[9] J. Mendling and M. Hafner. From inter-organizational workflows to process execution: Generating BPEL from WS-CDL. In *Proc. of OTM 2005, LNCS 3762*, 2005.

[10] Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[11] G. Salaun, L. Bordeaux, and M. Schaerf. Describing and reasoning on web services using process algebra. In *2nd Inter. Conference on Web Services.* IEEE, 2004.

[12] W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, N. Russell, H.M.W. Verbeek, and P. Wohed. Life after BPEL? In *Proc. of WS-FM 2005, LNCS 3670.* Springer, 2005.

[13] Xiangpeng Zhao, Hongli Yang, and Zongyan Qiu. Towards the formal model and verification of web services choreography description language. In *Proc. of WS-FM 2006, LNCS 4184.* Springer, 2006.

Appendix

Proof of Theorem 2:

In the first, we list some conventions:

- We will write $nproj(C_0, 1) \parallel ... \parallel nproj(C_0, n)$ as $p(C_0)$ for short.
- Before the discussion, we require any two activities of a Chore have different names except when they reside in two branches of a choice. Formally saying, in each substructure like "$C_1;\ C_2$" or "$C_1 \parallel C_2$", $Acts(C_1) \cap Acts(C_2) \subseteq \{\mathsf{skip}\}$. We can always rename some activities to achieve this condition.

We list some simple facts here:

Proposition 1 *If rp_i is role process $(i = 1, 2)$, then $[\![rp_1]\!] \frown [\![rp_2]\!] = [\![rp_1;\ rp_2]\!]$.*

Proposition 2 *If p_j^i is a role process, $p_1^1 \parallel p_1^2$ is free of dead lock, and so is $p_2^1 \parallel p_2^2$, then $[\![(p_1^1 \parallel p_1^2);\ (p_2^1 \parallel p_2^2)]\!] \subseteq [\![(p_1^1;\ p_2^1) \parallel (p_1^2;\ p_2^2)]\!]$.*

Definition 11 *Function actLead is defined as:*

$$\begin{array}{lcl} actLead(a_i) & \widehat{=} & \{a_i\} \\ actLead(p_1;\ p_2) & \widehat{=} & actLead(p_1) \\ actLead(p_1 \parallel p_2) & \widehat{=} & actLead(p_1) \cup actLead(p_2) \\ & & \cup\{c! \parallel c? \mid c! \in lead(p_i) \wedge c? \in lead(p_i)\} \end{array}$$

where the function lead is defined similar to the definition in **Sec. 4.2.1**, *but for the role process.* □

Proposition 3

$$[\![p]\!] = \bigcup_{t \in actLead(p)} [\![t;\ cutFirst(p, Acts(t))]\!]$$

when there is no choice in p *(*p *contains only ";" and "*$\|$*").* $cutFirst(p, A)$ *is a process* p' *which is obtained by erasing the first occurrence of each activity in* A *from the process* p*.*

For example, we have

$$\begin{aligned}&cutFirst((a_1;\ a_2), \{a_1\}) = a_2\\ &cutFirst((((c?;\ a_1) \parallel (c!;\ a_2));\ a_3), \{c!, c?\}) = (a_1 \parallel a_2);\ a_3\end{aligned}$$

Definition 12 *Restrict operator "*$\downharpoonleft$*" was applied on a role process and a set of process activities, resulting another role process. "*$\downharpoonleft$*" was defined recursively as below where ba stands for basic activity and* $\Box$ *stands for any of the connectors.*

$$\begin{aligned} ba \downharpoonleft S \ &\widehat{=} \begin{cases} ba & \text{if } ba \in S \\ skip & \text{if not} \end{cases} \\ (p_1 \Box p_2) \downharpoonleft S \ &\widehat{=}\ (p_1 \downharpoonleft S) \Box (p_2 \downharpoonleft S) \end{aligned}$$

$\Box$

The result of $p \downharpoonleft S$ may be simplified by equivalence laws to erase all unnecessary skip. This has no effect on the discussion below.

Definition 13 *We define a function* θ *from Chore activities to Role Process activity-set as follows:*

$$\begin{aligned} \theta(a_i) \ &\widehat{=}\ \{a_i\} \\ \theta(skip) \ &\widehat{=}\ \varnothing \\ \theta(i \xrightarrow{c} j) \ &\widehat{=}\ \{c?, c!\} \end{aligned}$$

we promote θ *to function on activity-set,* $\theta(S) = \bigcup_{p \in S} \theta(p)$*.* $\Box$

Lemma 1 *For any activity-set* ac*, for any Role Process* p*, if* p *is free of deadlock,* $[\![p]\!] \downharpoonleft ac = [\![p \downharpoonleft \theta(ac)]\!]$*.*

Proof: Transform p into Distributed Normal Form. Apply induction on number of connectors appeared in p.
Basics:
If p contains no connectors, p can not be $c!$ or $c?$ because p is free of deadlock by precondition, p must be a_i.

if $a_i \in ac$, then $a_i \in \theta(ac)$, then $[\![p]\!] \downharpoonleft ac = \{< a_i >\} = [\![a_i]\!] = [\![a_i \downharpoonleft \theta(ac)]\!]$, else $a_i \notin ac$,$a_i \notin \theta(ac)$,$[\![p]\!] \downharpoonleft ac = \{<>\} = [\![skip]\!] = [\![a_i \downharpoonleft \theta(ac)]\!]$.
Induction:
Suppose proposition is true if p contains less than k connectors, we are going to proof it also be true when p has exact k connectors.
Case 1: p is $p_1;\ p_2$,

$$\begin{aligned} [\![p_1;\ p_2]\!] \downharpoonleft ac &= ([\![p_1]\!] \frown [\![p_2]\!]) \downharpoonleft ac \\ &= [\![p_1]\!] \downharpoonleft ac \frown [\![p_2]\!] \downharpoonleft ac \\ &= [\![p_2 \downharpoonleft \theta(ac)]\!] \frown [\![p_2 \downharpoonleft \theta(ac)]\!] && \text{(hypothesis)} \\ &= [\![p_1 \downharpoonleft \theta(ac);\ p_2 \downharpoonleft \theta(ac)]\!] \\ &= [\![(p_1;\ p_2) \downharpoonleft \theta(ac)]\!] \end{aligned}$$

Case 2: p is $p_1 \sqcap p_2$,

$$\begin{aligned} [\![p_1 \sqcap p_2]\!] \downharpoonleft ac &= ([\![p_1]\!] \cup [\![p_2]\!]) \downharpoonleft ac \\ &= [\![p_1]\!] \downharpoonleft ac \cup [\![p_2]\!] \downharpoonleft ac \\ &= [\![p_2 \downharpoonleft \theta(ac)]\!] \cup [\![p_2 \downharpoonleft \theta(ac)]\!] && \text{(hypothesis)} \\ &= [\![p_1 \downharpoonleft \theta(ac) \sqcap p_2 \downharpoonleft \theta(ac)]\!] \\ &= [\![(p_1 \sqcap p_2) \downharpoonleft \theta(ac)]\!] \end{aligned}$$

Case 3: p is $p_1 \parallel p_2$,

$$\begin{aligned} & [\![p_1 \parallel p_2]\!] \downharpoonleft ac \\ &= (\textstyle\bigcup_{t \in actLead(p)} [\![t]\!] \frown [\![cutFirst(p, Acts(t))]\!]) \downharpoonleft ac && \text{(Proposition 3)} \\ &= \textstyle\bigcup_{t \in actLead(p)} ([\![t]\!] \downharpoonleft ac) \frown ([\![cutFirst(p, Acts(t))]\!] \downharpoonleft ac) \\ &= \textstyle\bigcup_{t \in actLead(p)} [\![t \downharpoonleft \theta(ac)]\!] \frown [\![cutFirst(p, Acts(t)) \downharpoonleft \theta(ac)]\!] \\ & && \text{(hypothesis)} \\ &= \textstyle\bigcup_{t \in actLead(p)} [\![t \downharpoonleft \theta(ac)]\!] \frown [\![cutFirst(p \downharpoonleft \theta(ac), Acts(t))]\!] \\ &= \textstyle\bigcup_{t \in actLead(p) \wedge [\![t]\!] \subseteq ac} [\![cutFirst(p \downharpoonleft \theta(ac), Acts(t))]\!] \\ &\quad \cup \textstyle\bigcup_{t \in actLead(p) \wedge [\![t]\!] \cap ac = \varnothing} [\![t]\!] \frown [\![cutFirst(p \downharpoonleft \theta(ac), Acts(t))]\!] \\ &= \textstyle\bigcup_{t \in actLead(p)} [\![(t;\ cutFirst(p \downharpoonleft \theta(ac), Acts(t)))]\!] \\ &= [\![p \downharpoonleft \theta(ac)]\!] && \text{(Proposition 3)} \end{aligned}$$

So, the lemma was proved. □

Lemma 2 *If C_i is RN ($i = 1, 2$), then $[\![p(C_1;\ C_2)]\!] \downharpoonleft Acts(C_i) = [\![p(C_i)]\!]$.*

Proof. Obviously, $p(C_1;\ C_2) \downharpoonleft \theta(Acts(C_i)) = p(C_i)$. By applying **Lemma 1**, we can get Lemma 2.

Lemma 3 *If C_i is RN ($i = 1, 2$), then for all $t \in [\![p(C_1;\ C_2)]\!]$, there exist $t_1 \in [\![C_1]\!], t_2 \in [\![C_2]\!]$, such that $t \in interleave(t_1, t_2)$.*

Proof. From lemma, we know $\forall\, t \in [\![p(C_1;\ C_2)]\!], \exists\, t_1 \in [\![p(C_1)]\!], t_2 \in [\![p(C_2)]\!]$, $s.t., t_1 = t \downharpoonleft Acts(C_1), t_2 = t \downharpoonleft Acts(C_2)$. Obviously, $Acts(t) \subseteq Acts(C_1;\ C_2)$ because C_i is RN and $p(C_i)$ is free of deadlock, so $Acts(t) = Acts(t_1) \cup Acts(t_2)$.

Now, we can construct t from t_1 and t_2 as following.

If $head(t) \in Acts(t_1)$, then $head(t) = head(t_1)$, otherwise $t \downharpoonleft Acts(t_1) \neq t_1$. The situation for t_2 is all the same. So we get the head of t_1, and recursively construct $tail(t)$ from $tail(t_1)$ and t_2. It's quit trivial and we omit the formal algorithm. □

We rewrite the two structural conditions here as a reference:

Condition 1. For each sub-structure as $A_1;\ A_2$, for each pair of $a \in end(A_1)$ and $b \in lead(A_2)$, there is a role r which is a performer of both a and b.

Condition 2. Suppose C is an n roles choreography, in its each sub-structure of the form $C_1 \sqcap, \ldots \sqcap C_m$, for each role number j except at most one role R^i, we all have $nproj(C_1, r^j) = \ldots = nproj(C_m, r^j)$ where $j \neq i$.

Lemma 4 *The conditions proposed above are sufficiency: When C is a choreography with n role, which satisfies* **Condition 1** *and* **Condition 2***, then it is a Restricted Natural Choreography.*

Proof. We prove it by induction on the structure of C.

Basics. When C is a basic activity, it is RN obviously.

Induction. According to structure of choreography, C must be $C_1;\ C_2$, or $C_1 \sqcap C_2$, or $C_1 \parallel C_2$. (For the structures with $\sqcap$ and $\parallel$, we prove here only the special cases with two branches. The general cases can be proved similarly but rather tedious.)

Case 1 (;): When C is $C_1;\ C_2$, where C_1 and C_2 are RN by inductive hypothesis. We have

$$\begin{aligned}
[\![C]\!] &= [\![C_1]\!] \frown [\![C_2]\!] && \text{(Definition)}\\
&= [\![p(C_1)]\!] \frown [\![p(C_2)]\!] && (C_1 \text{ and } C_2 \text{ are RN})\\
&= [\![p(C_1);\ p(C_2)]\!] && \text{(Proposition 1)}\\
&\subseteq [\![p(C_1;\ C_2)]\!] && \text{(Proposition 2)}\\
&= [\![p(C)]\!]
\end{aligned}$$

Thus $[\![C]\!] \subseteq [\![p(C)]\!]$. Now we are going to prove $[\![p(C)]\!] \subseteq [\![C]\!]$.

We know that for each $t \in [\![p(C)]\!]$, there exists $t_1 \in [\![C_1]\!]$ and $t_2 \in [\![C_2]\!]$ such that $t \in \mathit{interleave}(t_1, t_2)$ (Lemma 3). To prove that $t = t_1 \frown t_2$, we suppose it doesn't hold, then

$$\begin{aligned}
&\exists\, a_1 \in t_1, a_2 \in t_2 \bullet t = \langle ..., a_2, a_1, ...\rangle\\
&\Rightarrow \exists\, a_1' \in t_1, a_2' \in t_2 \bullet a_1' \in \mathit{end}(C_1) \wedge a_2' \in \mathit{lead}(C_2)\\
&\qquad\qquad\qquad\qquad \wedge\ t = \langle \ldots a_2', \ldots, a_1', \ldots\rangle\\
&\Rightarrow \exists\, i \bullet R^i \text{ performs both } a_1' \text{ and } a_2' \qquad\qquad \text{(Condition 1 holds in } C)\\
&\quad \wedge\ \mathit{nproj}(C, i) = \mathit{nproj}(C_1, i);\ \mathit{nproj}(C_2, i)\\
&\quad \wedge\ p(C) = \ldots \parallel \mathit{nproj}(C_1, i);\ \mathit{nproj}(C_2, i) \ldots\\
&\Rightarrow t = \langle \ldots, a_2', ..., a_1', \ldots\rangle \notin [\![p(C)]\!]
\end{aligned}$$

This conflict to the fact that $t \in [\![p(C)]\!]$. Thus we have

$$t = t_1 \frown t_2 \in [\![C_1]\!] \frown [\![C_2]\!] = [\![C]\!]$$

Therefore, we have $[\![p(C)]\!] \subseteq [\![C]\!]$ and $[\![C]\!] = [\![p(C)]\!]$.

Case 2 ($\sqcap$): Suppose C is $C_1 \sqcap C_2$ and $C_i (i = 1, 2)$ is RN. From **Condition 2**, we know, there is a role r^i, such that $\forall\, r^j, \mathit{nproj}(C_1, j) = \mathit{nproj}(C_2, j)$ when $r^j \neq r^i$. Thus we have

$$\begin{aligned}
&[\![p(C)]\!]\\
&= [\![(\mathit{nproj}(C_1, 1) \sqcap \mathit{nproj}(C_2, 1)) \parallel \ldots\\
&\quad \parallel (\mathit{nproj}(C_1, n) \sqcap \mathit{nproj}(C_2, n))]\!]
\end{aligned}$$

$$
\begin{aligned}
&= [\![nproj(C_1,1) \parallel \ldots \parallel nproj(C_1,i-1) \parallel (nproj(C_1,i) \sqcap nproj(C_2,i)) \\
&\quad \parallel nproj(C_1,i+1) \parallel \ldots \parallel nproj(C_1,n)]\!] && \text{(Condition 2)} \\
&= [\![nproj(C_1,1) \parallel \ldots \parallel nproj(C_1,i-1) \parallel (nproj(C_1,i) \sqcap nproj(C_2,i)) \\
&\quad \parallel nproj(C_1,i+1) \parallel \ldots \parallel nproj(C_1,n)]\!] && \text{(Definition)} \\
&= [\![p(C_1)]\!] \cup [\![nproj(C_1,1) \parallel \ldots \parallel nproj(C_1,i-1) \\
&\quad \parallel nproj(C_2,i) \parallel nproj(C_1,i+1) \parallel \ldots \parallel nproj(C_1,n)]\!] \\
&= [\![p(C_1)]\!] \cup [\![p(C_2)]\!] \\
&= [\![C_1]\!] \cup [\![C_2]\!] \\
&= [\![C]\!]
\end{aligned}
$$

Case 3 ($\parallel$): C is $C_1 \parallel C_2$ and C_1, C_2 are all RN. We have

$$
\begin{aligned}
[\![C]\!] &= interleave([\![C_1]\!], [\![C_2]\!]) && \text{(Definition)} \\
&= interleave([\![p(C_1)]\!], [\![p(C_2)]\!]) && (C_1, C_2 \text{ are RN}) \\
&= [\![p(C_1) \parallel p(C_2)]\!] && \text{(definition)} \\
&= [\![(nproj(C_1,1) \parallel nproj(C_2,1)) \parallel \ldots \\
&\quad\quad \parallel (nproj(C_1,n) \parallel nproj(C_2,n))]\!] && (\parallel \text{ is commutable}) \\
&= [\![nproj(C_1 \parallel C_2, 1) \parallel ... \parallel nproj(C_1 \parallel C_2, n)]\!] && \text{(definition)} \\
&= [\![p(C)]\!]
\end{aligned}
$$

The lemma is proved. □

Theorem 2 is a direct consequence of **Lemma 4**. If we transform the choreography into its Minimal Choice Normal Form, we can use **Condition 1** and **Condition 2** to find more Restricted Natural choreographies.